1 Introduction

The i.MX RT Series is industry’s first crossover processor provided by NXP. This document describes how to program a bootable image into the recovery Flash device and enable the i.MX RT to boot from recovery Flash device.

The ROM bootloader supports recovery boot - an option to recover the device to a certain state once the primary boot image is corrupted due to abnormal behavior, for example, a mistake which destroys the boot image during the image upgrade.

The release includes the PC-hosted blhost command-line application. This application is used for downloading application to Flash device in the development phase. This release also includes elftosb command-line application. It is used to generate bootable image for i.MXRT 600 ROM.

The software used for examples in this document is based on the i.MXRT 685 SDK 2.6.0. The development environment is IAR Embedded Workbench 8.40.2. The hardware development environment is X-IMXRT 685-EVK (Rev. E).

2 i.MXRT600 boot overview

2.1 Boot features

The internal ROM memory is used to store the boot code. After a reset, the Arm® processor starts its code execution from this memory. The boot loader code is executed every time when the part is powered-on or is reset.

Since the i.MX RT6 00 has no internal Flash for code and data storage, images must be stored elsewhere for loading upon reset or the CPU can execute from an external memory (XIP). Images can be loaded into on-chip SRAM from external Flash or downloaded via the serial ports (UART, SPI, I²C, USB). The code is then validated, and boot ROM will jump to on-chip SRAM.

Depending on the values of the OTP bits and ISP pins, and the image header type definition, the bootloader decides whether to download code into the on-chip SRAM or run from external memory. The bootloader checks the OTP bit settings first and then the ISP pins. If bit [3:0] in OTP word BOOT_CFG [0] is not programmed (4b'0000), the boot source is determined by the states of the ISP boot pins (PIO1_15, PIO1_16 and PIO1_17).

2.2 Boot settings

If PRIMARY_BOOT_SRC bits in OTP are not set, the i.MX RT6 00 will read the status of the ISP pins to determine the boot source.

Table 1. Boot mode and ISP downloader modes based on ISP pins

<table>
<thead>
<tr>
<th>Boot mode</th>
<th>ISP2 pin</th>
<th>ISP1 pin</th>
<th>ISP0 pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 1. Boot mode and ISP downloader modes based on ISP pins (continued)

<table>
<thead>
<tr>
<th>Boot mode</th>
<th>ISP2 pin PIO1_17</th>
<th>ISP1 pin PIO1_16</th>
<th>ISP0 pin PIO1_15</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDIO0 (SD Card)</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Boot from an SD card device connected to SDIO 0 interface. The i.MXRT600 will look for a valid image in the SD card device. If there is no valid image found, the i.MXRT600 will enter the ISP boot mode based on OTP <code>DEFAULT_ISP_MODE</code> bits (6:4, <code>BOOT_CFG[0]</code>)</td>
</tr>
<tr>
<td>FlexSPI Boot from Port B</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Boot from Quad or Octal SPI Flash devices connected to the FlexSPI interface 0 Port B. The i.MXRT600 will look for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the i.MXRT600 will enter ISP boot mode.</td>
</tr>
<tr>
<td>FlexSPI Boot from Port A</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Boot from Quad/Octal SPI Flash devices connected to the FlexSPI interface 0 Port A. The i.MXRT600 will look for a valid image in external Quad/Octal SPI Flash device. If there is no valid image found, the i.MXRT600 will enter ISP boot mode.</td>
</tr>
<tr>
<td>SDIO0 (eMMC)</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Boot from an SD card device connected to SDIO 0 interface. The i.MXRT600 will look for a valid image in the SD card device. If there is no valid image found, the i.MXRT600 will enter the ISP boot mode based on OTP <code>DEFAULT_ISP_MODE</code> bits (6:4, <code>BOOT_CFG[0]</code>)</td>
</tr>
<tr>
<td>USB DFU (master boot)</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>USB DFU class is used to download a boot image over the USB high-speed port into on-chip SRAM.</td>
</tr>
<tr>
<td>Serial ISP (UART, SPI, I²C, USB-HID)</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>The serial Interface (UART, SPI, and I²C, USB-HID) is used to program OTP, external Flash, SD or eMMC device</td>
</tr>
<tr>
<td>Serial master boot (UART, SPI, I²C, USB-HID)</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Serial master boot (SPI Slave, I²C Slave, or UART, USB-HID) is used to download a boot image over the serial interface (SPI Slave, I²C slave or UART, USB-HID)</td>
</tr>
</tbody>
</table>
2.3 Boot image offset

The bootloader looks for the boot image from a specified offset on a boot media. See the details in Table 2.

Table 2. Image offset on different boot media

<table>
<thead>
<tr>
<th>Boot media</th>
<th>Image offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlexSPI Boot (Serial NOR Flash device)</td>
<td>0x1000</td>
</tr>
<tr>
<td>SD Boot (SD card)</td>
<td>0x1000</td>
</tr>
<tr>
<td>eMMC boot (eMMC memory)</td>
<td>0x1000</td>
</tr>
<tr>
<td>Recovery Boot (SPI NOR Flash device)</td>
<td>0x1000</td>
</tr>
</tbody>
</table>

2.4 Boot image header

Once the boot mode is determined and the boot image is available on the selected external memory device (SD, eMMC or Serial NOR Flash), the ROM bootloader starts to copy the first 64 bytes of image header from the external memory device into on-chip SRAM. The beginning of the image follows the format mentioned in Table 3.

Table 3. Image Header format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00 - 0x1F</td>
<td>Reserved</td>
<td>—</td>
</tr>
<tr>
<td>0x20</td>
<td>imageLength</td>
<td>Image length</td>
</tr>
<tr>
<td>0x24</td>
<td>imageType</td>
<td>Image type</td>
</tr>
<tr>
<td></td>
<td>0x0000 - Plain image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0001 - Plain signed image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0002 - Plain CRC image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0003 - Encrypted signed image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0004 - Plain signed XIP image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x0005 - Plain CRC XIP image</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x8001 - Plain signed image with KeyStore included</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x8003 - Encrypted signed image with KeyStore included</td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>authBlockOffset/crcChecksum</td>
<td>Authenticate Block Offset or CRC32 checksum</td>
</tr>
<tr>
<td>0x2C - 0x33</td>
<td>Reserved</td>
<td>—</td>
</tr>
<tr>
<td>0x34</td>
<td>imageLoadAddress</td>
<td>Image load address</td>
</tr>
<tr>
<td>0x38-0x3F</td>
<td>Reserved</td>
<td>—</td>
</tr>
</tbody>
</table>

The bootloader begins scanning for user images by examining the image type located at offset 0x24 (imageType). If a valid image type is detected, the validation of an image header starts. Qualification of the image header continues by reading the image load address at offset 0x34 (imageLoadAddress) in the image header and using it as a pointer to a valid image header structure. If the imageType and imageLoadAddress are both non-zero, the address pointed by the imageLoadAddress must contain the image header under examination.

After the completion of the validation of the image header, the qualification continues by examining the image type field. If a bootable (not XIP) image resides in the external flash, the entire image will be loaded into the on-chip SRAM, and then the
2.5 Serial ISP boot

i.MXRT600 includes In-System Programming (ISP) functions. The bootloader provides flash programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond. Host-side command line and GUI tools are available to communicate with the bootloader. Users can utilize host tools to read and program application code and do manufacturing via the bootloader.

Here are brief ISP features:

- Supports UART, SPI, I2C and USB peripheral interfaces.
- Automatic detection of the active peripheral.
- Programming OTP.
- Programming Serial NOR Flash.
- Programming SD Card.
- Programming eMMC device.

Each kind of boot device has unique config option block. It indicates the flash device properties. The config option block should be passed to host tool. Table 4 describes the configuration option block for recovery boot device.

Table 4. Recovery boot configuration option block

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Option</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[31:28] Tag</td>
<td>Must be 0xC</td>
</tr>
<tr>
<td></td>
<td>[27:24] Reserved</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td></td>
<td>[23:20] SPI Index</td>
<td>Flexcomm SPI Index Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000 - SPI0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0001 - SPI1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010 - SPI2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0011 - SPI3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0100 - SPI4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0101 - SPI5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110 - SPI6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0111 - SPI7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Others reserved</td>
</tr>
<tr>
<td></td>
<td>[19:16] Reserved</td>
<td>This field is reserved.</td>
</tr>
<tr>
<td></td>
<td>[15:12] Memory Type</td>
<td>Memory Type Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0000 - Manual NOR Flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0010 - SFDP NOR Flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Others reserved</td>
</tr>
</tbody>
</table>

Table continues on the next page...
Table 4. Recovery boot configuration option block (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
|        | [11:8] Memory Size | 0 <= n <= 11, size = 512 KBytes * 2^n  
               |                      | 12 <= n <= 15, size = 32 KBytes * 2^(n-12)               |
|        | [7:4] Sector Size    | 0 <= n <= 1, size = 4 KBytes * 2^n  
               |                      | 2 <= n <= 5, size = 32 KBytes * 2^(n-2)  
               |                      | Others reserved     |
|        | [3:0] Page Size      | 0 <= n <= 2, size = 256 Bytes * 2^n  
               |                      | 3 <= n <= 5, size = 32 Bytes * 2^(n-3)   
               |                      | Others reserved     |

3  Recovery boot mode

In i.MXRT600, the SPI NOR device is supported as the recovery media.

3.1 Design pin assignment

One of the SPI0-SPI7 can be specified as the interface to connect an SPI NOR device.

Table 5 describes the pin list of the SPI peripheral.

Table 5. Recovery boot SPI flash pin assignments

<table>
<thead>
<tr>
<th>Boot interface</th>
<th>Pin(s)</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI flash</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC0</td>
<td>Use Flexcomm0 pins PIO0_0 (SCK), PIO0_1 (MISO), PIO0_2 (MOSI), PIO0_3 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC1</td>
<td>Use Flexcomm1 pins PIO0_7 (SCK), PIO0_8 (MISO), PIO0_9 (MOSI), PIO0_10 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC2</td>
<td>Use Flexcomm2 pins PIO0_14 (SCK), PIO0_15 (MISO), PIO0_16 (MOSI), PIO0_17 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC3</td>
<td>Use Flexcomm3 pins PIO0_21 (SCK), PIO0_22 (MISO), PIO0_23 (MOSI), PIO0_24 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC4</td>
<td>Use Flexcomm4 pins PIO0_28 (SCK), PIO0_29 (MISO), PIO0_30 (MOSI), PIO0_31 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC5</td>
<td>Use Flexcomm5 pins PIO1_3 (SCK), PIO1_4 (MISO), PIO1_5 (MOSI), PIO1_6 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC6</td>
<td>Use Flexcomm6 pins PIO3_25 (SCK), PIO3_26 (MISO), PIO3_27 (MOSI), PIO3_28 (SSEL)</td>
<td></td>
</tr>
<tr>
<td>FC7</td>
<td>Use Flexcomm7 pins PIO4_0 (SCK), PIO4_1 (MISO), PIO4_2 (MOSI), PIO4_3 (SSEL)</td>
<td></td>
</tr>
</tbody>
</table>

3.2 Recovery boot flow

The bootloader enters the recovery boot mode if the master boot fails and the recovery boot is enabled.
When the recovery boot process starts, the bootloader probes the presence of the SPI NOR device by checking the manufacturer ID, using 24 MHz clock from IRC48M. Once detected, the bootloader stays loading the recovery image from the SPI NOR device to the on-chip SRAM, uses the 24 MHz clock, and performs the integrity check/image authentication with the image.

The bootloader jumps to the recovery boot image if the integrity check/authentication passes. Otherwise, it falls through to the ISP mode.

**3.3 Image link region**

Recovery image can only be Non-XIP image. It should be linked into internal 4.5 MB SRAM. As the first 512 KB SRAM has been occupied by ROM and DSP, it is better to link Recovery image from 0x80000.
3.4 Recovery boot OTP settings

There are two recovery boot related fields. Both of these two fields are allocated at `BOOT_CFG0` OTP word.

- **PRIMARY_BOOT_SRC**, starting from offset 0, 4-bit width. See the details in Table 6.

**Table 6. Recovery boot OTP field**

<table>
<thead>
<tr>
<th>PRIMARY_BOOT_SRC</th>
<th>Field</th>
<th>Primary boot Source. (a.k.a. Master boot source)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI _SLV_BOOT</td>
<td>b'0111</td>
<td>Boot from 1 bit NOR flash via SPI interface, The SPI instance used is chosen by fuse word 0x60 bit17 to bit 19, more details please refer fuse map.</td>
</tr>
<tr>
<td>FLEX SPI_REC_BOOT _PORTB</td>
<td>b'1 011</td>
<td>Boot from Octal/Quad SPI flash device on FlexSPI0 Port B. If the image is not found check recovery boot using SPI-flash device through FlexComm.</td>
</tr>
<tr>
<td>FLEX SPI_REC_BOOT _PORTA</td>
<td>b'1100</td>
<td>Boot from Octal/Quad SPI flash device on FlexSPI0 Port A. If the image is not found check recovery boot using SPI-flash device through FlexComm.</td>
</tr>
<tr>
<td>SDHC0_REC_BOOT</td>
<td>b'1101</td>
<td>Boot from SDHC0 port device. If image is not found check recovery boot using SPI-flash device through FlexComm.</td>
</tr>
<tr>
<td>SDHC1_REC_BOOT</td>
<td>b'1110</td>
<td>Boot from SDHC1 port device. If image is not found check recovery boot using SPI-flash device through FlexComm.</td>
</tr>
</tbody>
</table>

- **REDUNDANT_SPI_PORT**, starting from offset 17, 3-bit width. See the details in Table 7.
## Table 7. Recovery boot OTP field

<table>
<thead>
<tr>
<th>REDUNDANT_SPI_PORT</th>
<th>FlexComm port to use for redundant SPI flash boot</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC0</td>
<td>Use Flexcomm0 pins PIO0_0 (SCK), PIO0_1 (MISO), PIO0_2 (MOSI), PIO0_3 (SSEL)</td>
<td>3'b000</td>
</tr>
<tr>
<td>FC1</td>
<td>Use Flexcomm1 pins PIO0_7 (SCK), PIO0_8 (MISO), PIO0_9 (MOSI), PIO0_10 (SSEL)</td>
<td>3'b001</td>
</tr>
<tr>
<td>FC2</td>
<td>Use Flexcomm2 pins PIO0_14 (SCK), PIO0_15 (MISO), PIO0_16 (MOSI), PIO0_17 (SSEL)</td>
<td>3'b010</td>
</tr>
<tr>
<td>FC3</td>
<td>Use Flexcomm3 pins PIO0_21 (SCK), PIO0_22 (MISO), PIO0_23 (MOSI), PIO0_24 (SSEL)</td>
<td>3'b011</td>
</tr>
<tr>
<td>FC4</td>
<td>Use Flexcomm4 pins PIO0_28 (SCK), PIO0_29 (MISO), PIO0_30 (MOSI), PIO0_31 (SSEL)</td>
<td>3'b100</td>
</tr>
<tr>
<td>FC5</td>
<td>Use Flexcomm5 pins PIO1_3 (SCK), PIO1_4 (MISO), PIO1_5 (MOSI), PIO1_6 (SSEL)</td>
<td>3'b101</td>
</tr>
<tr>
<td>FC6</td>
<td>Use Flexcomm6 pins PIO3_25 (SCK), PIO3_26 (MISO), PIO3_27 (MOSI), PIO3_28 (SSEL)</td>
<td>3'b110</td>
</tr>
<tr>
<td>FC7</td>
<td>Use Flexcomm7 pins P4_0 (SCK), P4_1 (MISO), P4_2 (MOSI), P4_3 (SSEL)</td>
<td>3'b111</td>
</tr>
</tbody>
</table>

## 4 MIMXRT685 EVK board settings

There is no Flash connected to Flexcomm SPI port on the EVK board. To enable the Recovery QSPI NOR Flash boot feature, a simple Flash memory card is needed.

![QSPI NOR Flash memory card](image)

**Figure 3. QSPI NOR Flash memory card**

It is QuadSPI NOR Flash in the memory card. However, if it is used as recovery boot device, only IO[1:0], CLK, and CS pins are needed to connect to Flexcomm SPI port. Other IO[3:2] pins should be driven to high.

Flexcomm SPI5 (J28 pin3-6) is selected to connect to Flash memory card.
As the QSPI NOR Flash chip in memory card is powered by 3.3 V, to make sure that the i.MXRT600 GPIO can drive this flash, JP12 pin2-3 should be connected.

5 Program tools

5.1 blhost tool

The blhost is a command-line host program used to interface with devices running ROM Bootloader. The version of blhost should be v2.3 or higher.

5.2 NXP-MCUBootUtility tool

The NXP-MCUBootUtility is a GUI tool used to interface with devices running ROM Bootloader. It is a real one-stop tool. The version of NXP-MCUBootUtility should be v2.2 or higher.
5.3 Use blhost to enable recovery boot

This chapter shows the steps to use blhost tool to program an image to QSPI NOR Flash and boot from the Recovery QSPI NOR Flash.

1. Open the `\SDK_2.6.0_EVK-MIMXRT685\boards\evkmimxrt685\driver_examples\gpio\led_output` example and select the project configuration as debug, as shown in Figure 6.

   ![Figure 6. Selecting the project configuration as debug](image)

2. Build the project and generate an image with `.bin` format. You can find `gpio_led_output.bin` as shown in Figure 7.

   ![Figure 7](image)
3. In `startup_MIMXRT685S_cm33.s`, fill actual image length according to the size of generated `gpio_led_output.bin`. Rebuild the project to get new `gpio_led_output.bin`. 
4. Copy the new `gpio_led_output.bin` to the `blhost` folder, as shown in Figure 9.

5. Switch the RT685-EVK board to Serial ISP mode by setting SW5 to 1-ON, 2-OFF, and 3-OFF. Connect a USB cable to J7 USB port and issue the `blhost` commands, as shown in Figure 10.
The argument value 0xc0500000 in the fill-memory command is recovery boot config option block. See Table 4 for details.

6. Issue efuse-program-once 0x60 recoveryBootValue to set PRIMARY_BOOT_SRC bits in OTP BOOT_CFG [0]. recoveryBootValue could be 4b’0111/4b’1011/4b’1100/4b’1101/4b’1110. Make sure that there is no valid image in any master boot device. Reset the board and the gpio led demo will run properly.

5.4 Use NXP-MCUBootUtility to enable recovery boot

This chapter shows the steps to use blhost tool to program an image to QSPI NOR Flash and Boot from the Recovery QSPI NOR Flash.

1. Rebuild \SDK_2.6.0_EVK-MIMXRT685\boards\evkmimxrt685\driver_examples\gpio\led_output project and generate an image with the .srec format.

2. Switch the RT685-EVK board to Serial ISP mode, connect a USB cable to J7 USB port, and then open NXP-MCUBootUtility. Set the MCU device to iMXRT6xx and Boot Device to FLEXCOMM SPI NOR. Click Boot Device Configuration to set Spi Index to 5. Click Connect to ROM.
3. If the tool can connect with RT600 BootROM successfully, the device information will be shown on the Device Status pane. Browse the gpio_led_output.srec file and click All-In-One Action.
4. Burn the `recoveryBootValue` to set `PRIMARY_BOOT_SRC` bits in OTP `BOOT_CFG[0]`. `recoveryBootValue` could be 4b'0111/4b'1011/4b'1100/4b'1101/4b'1110.
5. Make sure that there is no valid image in any master boot device. Reset the board and the gpio led demo will run properly.
Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniIV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Converge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphomy, VortiQa, Hybird, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, μVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2020. All rights reserved.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 25/02/2020
Document identifier: AN12751